Boolean expressions

• A boolean expression is an expression that is either 'True' or 'False'

```python
>>> 5 == 5
True
```

```python
>>> 5 == 6
False
```

```python
>>> 5 == 5
True
```

```python
>>> 5 == 6
False
```
Boolean expressions

- The keywords 'True' and 'False' are special values of type bool

- They are not strings!

```python
>>> type(True)
<type 'bool'>
```
Relational operators

- The equality operator '==' is just one of the relational operators:
  - Inequality: '!='
  - Greater than: '>
  - Less than: '<'
  - Greater or equal: '>='
  - Less or equal: '<='
Logical operators

- There are three logical operators:
  - 'and'
  - 'or'
  - 'not'

- They are all keywords
Logical operators

- In order for an 'and' expression to be 'True' both of its operands must be 'True'

- In order for an 'or' expression to be 'True' either of its operands may be 'True'

- The 'not' operator negates a boolean expression
Conditional execution

• An 'if' statement gives us the ability to make decisions:

```python
if x > 0:
    print('x is positive')
```

• If the condition 'x>0' is 'True' the print will execute
Conditional execution

- The boolean expression after the 'if' is called the condition

- If the condition evaluates to 'True' the indented statement gets executed, otherwise it does not

- There can be more than one indented statement
Alternative execution

• An 'if...else' statement gives us two possibilities for execution controlled by the boolean expression:

```python
if x % 2 == 0:
    print('x is even')
else:
    print('x is odd')
```
Alternative execution

- If the condition evaluates to 'True' then the statement in the 'if' branch is executed

- If the condition evaluates to 'False' then the statement in the 'else' branch is executed

- Since the statement must be either 'True' or 'False' exactly one statement will be executed
Chained conditionals

- Sometimes there are more than two possibilities and we need more than two branches.

- Adding an 'elif' clause to an 'if' or 'if...else' statement gives us as many branches as we need.
Chained conditionals

• An example of an 'if...elif...else' statement:

```python
if x < y:
    print('x is less than y')
elif x > y:
    print('x is greater than y')
else:
    print('x is equal to y')
```
Chained conditionals

- 'elif' is an abbreviation of “else if”
- Still only one branch gets executed
- There is no limit on the number of 'elif' clauses, but there can be only one (or none) 'else' branches
Chained conditionals

- Each branch is checked in order

- If the first conditional is 'False' then the second branch gets checked, and so on

- The first branch whose conditional evaluates to 'True' gets executed
Chained conditionals

• Even if more than one condition is 'True' only the first true branch executes

• Here are the rules for 'if' statements:
  • First, exactly one 'if' clause with condition
  • Second, zero or more 'elif' clauses with condition
  • Third, zero or one 'else' clause with no condition
Nested conditionals

- Any type of statements can be indented inside of an 'if' statement; even other 'if' statements!

```python
if x == y:
    print('x is equal to y')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```
Nested conditionals

- In the example the outer 'if' has two branches

- The outer 'else' clause has an 'if...else' statement

- Proper indentation is critical
Nested conditionals

- Sometimes logical operators can be used instead:

  ```python
  if x > 0:
      if x < 10:
          print('x is greater than zero, but less than 10')
  ```

- Gets re-written as:

  ```python
  if x > 0 and x < 10:
      print('x is greater than zero, but less than 10')
  ```